Supernetworks

Peter J. Denning

October 31, 1984
Revised January 16, 1985

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS TR 84.10

Research Institute for Advanced Computer Science

Supernetworks

Peter J. Denning

Research Institute for Advanced Computer Science
NASA Ames Research Center

RIACS TR 84.10
October 31, 1984
Revised January 16, 1985

Future computer networks, such as Sciencenet, will be heterogeneous -- composed of many subnets of different characteristics. Yet these networks must provide simple, uniform access to community resources. This paper explores five principles for network architecture in pursuit of this goal.

An early version of this paper was prepared as a position statement presented to a Sciencenet planning committee meeting on October 26, 1984.

Work reported herein was supported in part by Contract NAS2-11530 from the National Aeronautics and Space Administration (NASA) to the Universities Space Research Association (USRA).

Supernetworks

Peter J. Denning
Research Insitute for Advanced Computer Science

October 31, 1984
Revised January 16, 1985

1. BACKGROUND

A computer netwok is a collection of computers, called "hosts," that can communicate with one another. A host can be a large supercomputer, a time-shared minicomputer, or a personal workstation. Ordinary terminals are not considered hosts.

I am interested here in a new class of networks herein called "supernetworks." They are complex, heterogeneous systems built up from many subnetworks of widely differing types. Yet supernetworks must provide simple, uniform access to community resources.

It is critically important that the architecture of supernetwork be based on a small set of solid principles aiming at a usable, reliable, and robust system.

The principles must be flexible enough that the network can evolve smoothly as the needs of the community change. I will describe here a model comprising five principles that meet this goal. This model focuses on how the network looks to its users rather than how it looks to its designers. In the terminology of the ISO model, this paper will focus on requirements to be met at the Applications Layer.

I offer this model as a way of thinking about Sciencenet, the national network for the scientific research community being planned by the National Science Foundation (NSF).

2. COMPONENTS

There are four major types of objects in the architecture of a large computer network:

1. **Resources** are computer-based facilities accessible to members of the user community. Resources are based in hosts maintained by member institutions (and sometimes by individuals). In Sciencenet, the initial resources will be supercomputer centers. In time the resources will also include program and data libraries, services, and special purpose facilities.

2. **Connections** are reliable communications paths linking users to the required resources. A connection provides access to a resource. Connections must be potentially capable of enabling a remote user to obtain the same level of service as is available to local users. Connections should hide variations among different communications media -- e.g., bandwidth or delay. (Connections are implemented by the Transport or Session Layers of the ISO network protocol model.)

3. **Functions** are operations provided by the network; they allow users to cause the network to do work for them.

4. **Names** are identifiers used to address resources and individual users. Generally there will be two types of names. *External names* are character strings that have meaning to users. *Internal names* are binary codes that allow network hardware and software to efficiently address whatever is named. The network software (not the users!) must be responsible for translating external to internal names.

The fourth component -- names -- is perhaps the most subtle and most difficult component. An inadequate naming scheme can seriously complicate the network command interface and make the process of changing the names or locations of resources very expensive.

3. PRINCIPLES

The network architecture should be guided by five principles listed below and considered in detail in the sections following. These principles enable attaining the goal of a usable, reliable, robust, and evolvable network.

1. *Resources may be things other than supercomputer centers* because scientific program and data libraries, services, and unique facilities should all be sharable throughout the community.

2. *The network should be people-oriented, not just facility-oriented.* Scientists should be able to exchange messages and files among themselves. The network should encourage collaboration and sharing and not merely provide connections to computers. Put another way, the scientists in the community should be regarded as (human) resources to which access must be provided.

3. *A connection should be regarded only as a communication path* from one point to another. The speed of the connection and the physical networks it may traverse should be hidden from all users except those who wish to see such details. The basic functions for establishing and using connections -- i.e., the command interface -- should be simple and independent of the physical media through which connections are made.

4. *The network communications interface should be simple and locally extensible.* Every host's operating system will contain software that implements a standard base set of network commands and manages communication with the physical network to which the host is attached. The standard base set of commands should be simple because it may serve as the user interface on some hosts. Each host should, however, have the option to extend the base set of commands to meet local needs.

5. *User and resource names should be location independent.* The system of (external) names for resources should be carefully designed from the outset and seldom if ever changed. The network, not the users, should be responsible for keeping track of the current locations of users and network resources. Users should not be required to know the geographic locations of resources in order to access them. A change in the location of a user or a resource should not require editing or recompiling any files or programs outside the network database.

A summary of these principles is: The network should provide uniform access to its resources and users without requiring users to know the physical locations of resources or other users. The base commands for evoking actions from the network and the names for addressing resources or users should be the same everywhere in the network. The manner of using a connection should not

depend on the physical path or the speed -- the path should be selected automatically to by the network to minimize cost and the speed should be whatever the user is willing to pay for.

These principles should be regarded as guidelines, not hard rules. Compromises may be necessary along the way. The implementation of the network should use these guidelines to the maximum extent possible.

3.1. Principle 1 -- Generality of Resources

The first resources on Sciencenet will be supercomputer centers. This is because a priority of the supercomputing program is to provide advanced computational power to the members of the scientific community.

In time, however, the community will begin to accumulate new knowledge from results obtained by advanced computation. The new knowledge should become accessible and usable by the community. This can be accomplished by allowing (and encouraging) new computational resources attached to the network. Examples include:

Program libraries	--	For example, software for image processing, statistical analyses, or graphics support.
Data Libraries	--	For example, raw or condensed data from experiments or satellite instruments, partially processed images, catalogs of new astronomical objects, or catalogs of chemical compounds.
Services	--	For example, software distribution services, information services that aid locating people or data of special kinds, and directories of users or other services.
Facilities	--	For example, a center offering special processing and expertise in a discipline (e.g., NASA Ames's NAS for fluid dynamic computations), a real-time satellite daa collector, or a 3-D graphics processor.

This list emphasizes resources that are *producers* of information. Each producer is represented by a computer node attached to the network. In contrast, users are primarily *consumers* of information. They are also represented by computers -- typically their host institution's time-sharing computer or by personal workstations. The "generality of resources" principle says that all these distinctions -- producers versus consumers, various types of producers, various types of consumers -- should be ignored in the network architecture. The network should be designed to establish and maintain communications among computers without having to know the functions of those computers.

If the only type of facility provided for in the network design is a supercomputer center, the network may be locked out of a development path toward these other types of resources when the community starts generating them.

3.2. Principle 2 -- People Orientation

The network should be people-oriented. It should make resource access and communication among users convenient and straightforward. This is so for at least two reasons. First, collaboration and sharing are the most powerful methods known for advancing scientific knowledge. Second, the network should encourage the development of large pools of young people skilled in computing technology.

Another way of saying this is that the second use of the word "network" in English is important: a network is a group of people, perhaps widely distributed, bonded together toward a common goal.

People-orientation has many implications for the design of network software at all levels. For example:

1. In addition to internal names for resources (users, files, and facilities) easily processed by software and hardware, a system of external names meaningful to users is required.

2. Because names for resources will be embedded in programs, files, and directories, names must be location independent. This is so that if the physical location of a resource changes, none of the programs, files, or directories that refer to it need be edited or recompiled.

3. The primary mechanism of interaction among users will be electronic mail.

These implications will be explored further below, mostly in the section on location independence. The mail system illustrates most of the problems that can arise if names are location dependent. These problems are worse for files and facilities than for users.

3.3. Principle 3 -- Connection is a Path

There is a potential for much confusion among the multiplicities of networks, protocols, bandwidths, transmission modes, media, routings, gateways, addressing schemes, and operating systems. Most of these factors are physical details that need not concern any user except the few experts who wish to see them.

The most effective way of hiding these details is to regard a connection merely as a transmission path between two nodes of the network. The two ends

of a connection are called "sockets". Devices, files, and executing programs can be connected interchangeably to sockets.

The network software will set up the path over one or more networks that physically connect the two nodes between which connection is desired. The network software will be responsible for routing, error control, protocol selection, gateway selection, generation of internal names recognizable by the physical networks, etc. The user will not be required to be aware of these choices.

A user can control factors such as delay and bandwidth of a connection by stating how much he is willing to pay. If a user is willing to pay only for a 9600 baud connection, the network software should automatically provide this level of service. A user willing to pay the higher cost of satellites should automtically be given that grade of service (after the hardware is installed). A user should, of course, be able to inquire what speed or cost connection he has, but he should not be required to issue a different set of commands for each type of connection available.

This principle -- *distinguishing a logical connection from a physical connection* -- is essential to keep the user interface simple and to allow upgrades to future types of service without requiring new commands to be defined.

Obviously, scrupulous application of this principle cannot hide *all* the details. It can certainly hide the working details of commands to open, close, send, or receive -- but it cannot hide measurable properties such as bandwidth or response time. Nonetheless, a surprising amount of detail can be suppressed by

this principle.

3.4. Principle 4 -- Simple Basic but Extensible Interface

The *network interface* is the set of commands that evoke actions from the network. The essential ones are:

Open a connection with another node.
Listen for an incoming call.
Close a connection.
Attach a file, device, or program to a socket.
Detach a file, device, or program from a socket.
Send data into an open connection.
Receive data from an open connection.
Find out the status of a connection.
Send a file or mail to a user.
Retrieve a file from another node.
Update the database containing user and resource names.

An interface this simple can provide most of the function we expect of a network. The first eight commands manipulate and use connections; they will be illustrated in the example below. The open and close commands must enforce the policies of access control of the local computer and of the network.

The command to send (or mail) a file to a user causes a named file to be sent to a directory (or mailbox) on the user's home computer. This command is automatically used by a user-level mail program to send mail. On being asked to receive mail, the user-level mail program retrieves directly from the equesting user's local mailbox; hence no special network command is needed to receive

mail.

The command to retrieve a file copies a file from from a remote computer to the local computer without requiring the user to log in remotely. This command will, however, normally request the user to provide a password.

The commands to manipulate the network users' database allow records showing the location of each user and resource to be kept current. They also allow one user to locate others by name, research area, job title, host computer, etc. Database query commands can be issued automatically by user-level mail programs so that mail will be delivered to the correct destination even if the addressee moves.

3.4.1. Example

Consider a session in which the user wishes to start a job on a remote supercomputer with input from a local file; he wishes to control the job from his terminal and connect its output to a special high-speed display. The steps (illustrated in Figure 1) are:

1. Open an outbound connection (A) and an inbound connection (B) with the supercomputer node. By default, the supercomputer will attach a command interpreter program to the other end of the outbound connection and will attach its output to the other end of the inbound connection.

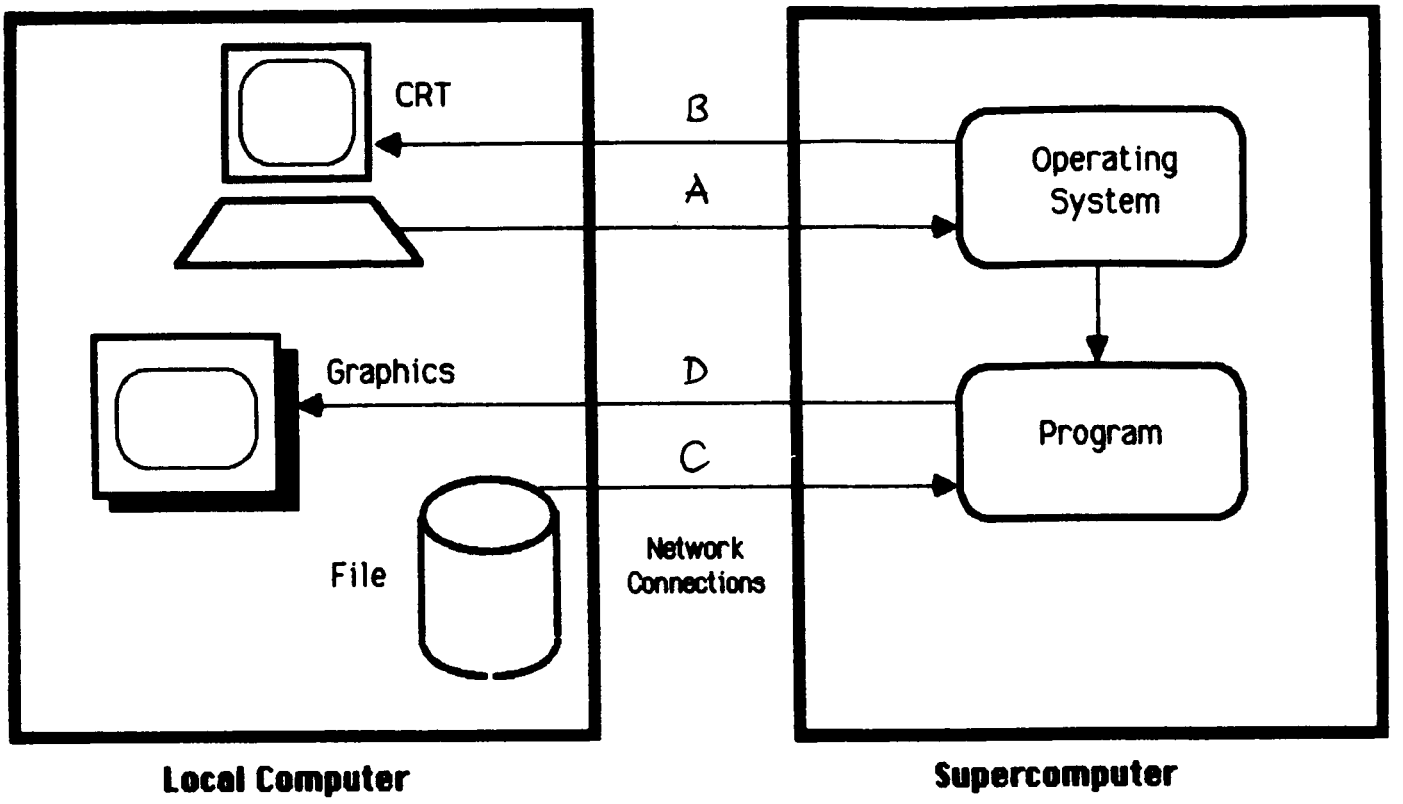

FIGURE 1. Connections between local computer and a supercomputer. Connections A-B is for control and connections C-D is for program I/O.

2. Attach the local keyboard to the outbound connection and the local terminal display (CRT) to the inbound connection.

3. Type the commands needed to log in to the operating system on the supercomputer.

4. Type the commands needed to spawn a process on the supercomputer containing the program to be executed there.

5. Open a second outbound connection (C) to the supercomputer node and (using connection pair AB) command the supercomputer operating system to attach the remote program's input to the other end of that connection.

6. Open a second inbound connection (D) to the supercomputer node and (using connection pair AB) command the supercomputer operating system to attach the remote program's output to the other end of that connection.

7. Locally attach the input file to the outbound connection (C) and the graphics display to the inbound connection (D).

8. Command the supercomputer operating system to start the remote program. Issue further commands as necessary to control that program. (These interactions occur over connection pair AB.)

9. When the remote program has completed, close down all the above connections.

The above pattern of commands is likely to be so common that a command script can be written embodying them. The user then need only execute the command script with a few parameters to describe the remote program and the local sources of input and output. For example, the command script can have the name RE (for "remote execute") and calling form

RE(node_name, program_name, input, output).

The configuration of Figure 1 can then be established with the specific call

RE(Supercomputer, Program, File, Graphics).

Note how a command script can hide the network command interface behind a simpler, higher-level interface.

With a sufficiently powerful local operating system, most of the above steps can be executed implicitly by the local shell (command interpreter), which would hide even more of the network interface. For example, one can envisage a modification of UNIX allowing the user to type

Program < File > Graphics &

This command would initiate execution of the named "Program" with its input directed from the named "File" and output directed to the named "Graphics" display. The symbol "&" means that the command will be executed in the background; the user's terminal remains connected to the controlling operating sys-

tem in the foreground. Associated with the local name "Program" is a link to the actual program on the supercomputer node: hence the attempt to execute it will automatically cause the shell to set up the configuration shown in Figure 1.

3.4.2. Extensions

This example illustrates that a simple network command interface would allow the network to be used with a wide variety of operating systems.

1. An operating system with no network support can simply let the network interface be available directly to users. Because the interface is simple, this presents no problem except that some tasks (such as remote job execution) can become tedious.

2. An operating system whose shell accommodates macro commands can easily be extended with command scripts for common network operations (such as remote job execution). In this case, some of the network interface becomes hidden behind command scripts.

3. An operating system with a sufficiently powerful shell can be extended by having the shell issue the network commands on discovering that command components are not local. In this case, almost all the net-

work interface can be hidden.

It is important that the basic interface be simple. If the design of a network is approached without a uniform, usable, basic interface, there is a serious danger the set of network commands can become quite large. The Sciencenet planning documents, for example, discuss the need for "high-level protocols" for file transfer, terminal emulation, remote job entry (illustrated above), interactive graphics, and electronic mail. Without a base of common concepts for connections and their use, these programs will collectively constitute a very complicated and difficult user interface.

3.5. Principle 5 -- Location Independent Names

Location independence is the property that external names are interpretable by the network no matter what the physical locations of the objects or persons denoted. This principle is enforced partly by network administration and partly by designing software to distinguish name from location. Users of networks with this property can be assured that commands and programs will continue to work even if files or users should move.

Network databases play prominent roles in location independence. These databases maintain records of the correspondences between external names and internal names. A prime example of such a database is the directory of users,

often called a *nameserver*. Directory servers for software, files, and special facilities can also be set up so that network users can locate these objects knowing only their (external) names or their functions.

The electronic mail system illustrates most of the benefits of location independent names. Today, someone wishing to send me mail via ARPANET must know my local login name (pjd) and my host computer name (riacs): the command to send me mail is

Mail pjd@riacs

If I change my account to another host computer, say "Ames," everyone must now use the command

Mail pjd@Ames

The question is, how does everyone find out? In the ARPANET, I must notify the Network Information Center (NIC). A few days later, users can manually interrogate the NIC database to discover my new address.

Many mailing programs permit users to define their own private *alias tables* that define correspondences between the user's nicknames and proper network addresses. Thus one user can say "alias peter = pjd@riacs" while another can say "alias riacs_director = pjd@riacs". The mailer will thereafter correctly interpret the command "Mail peter" by the first user and "Mail riacs_director" by the second. In case I change my login name or my host computer, this

scheme can cause problems because a location-dependent string ("pjd@riacs") is hidden in a table. The problem of updating these tables can become serious when location-dependent strings are hidden in distribution lists, where the person controlling the list is different from any of the list's users.

These complications disappear with location-independent names. This could be enforced administratively by having the NIC assign names that are unique during the expected lifetime of the network. For example, the string "peter_denning" can be assigned as my fixed, permanent network name. Mailers must be redesigned to query the NIC nameserver for the current ddress associated with a name. Thus if someone says

Mail peter_denning

his mailer program will automatically query the NIC nameserver and will discover that "peter_denning" currently has network address "pjd@riacs". If I change my host computer to "Ames", I merely notify the NIC. All alias tables and mailing lists can contain the string "peter_denning" and will work correctly after the NIC has recorded my new address.

Central administrative over the entire nameserver database can become expensive and cumbersome in a large network. A more flexible solution allocates responsibilities as follows:

1. The network administration retains control over the assignment of name-strings to resources, institutions, and users (thereby assuring uniqueness):

2. Insitutions control accounts on their machines; and

3. Users control other fields of their entries in the network database.

Under this scheme, a user's identifier string would be assigned by the NIC; the user's institution would assign a mailbox name to the appropriate field of the user's record in the NIC database; and the user would update job titles, keywords indicating research areas, etc. A scheme of this type is used by CSNET.*

With a nameserver database, users (and resources) can be located by descriptive keywords. For example, the keyword pairs

peter denning
riacs director

*This is not the only way of delegating responsibility over portions of the nameserver database. It is also possible to recognize administrative entities such as agencies or disciplines so that addresses like "denning@computer_science" or "denning@NASA-Ames" would be meaningful.

would uniquely identify my record in the nameserver database; my record contains my current network address. The command "Mail peter denning" would work as long as I am a registered user whereas "Mail riacs director" would work only until I changed jobs.

The extension of mailer programs to deal with location independent names (or with keywords) has not been done in the software generally available in the ARPANET community. It is available in CSNET.

The location problem is more serious with addressing of other network resources, such as files and facilities, than with persons. This is because file and facility names are normally included in programs. If the location of the file or facility should change, all programs that refer to it will no longer work correctly and will require relinking or recompiling. This can be very expensive if a popular file is moved in a large network.

4. OTHER CONSIDERATIONS

A good administrative structure is important for a supernetwork. The network administration must provide ready assistance for user questions and problems of many kinds -- helping new institutions come on line, answering questions about problems in the software, supporting mailing lists and bulletin boards, maintaining network software, developing new services, setting connection standards for new servers that might be added, negotiating rates with public carriers,

and the like. The network administration should provide for considerable input from the user community to help with questions such as: What new services or improvements do users want? How does a group with a new resource to offer go about making that resource available to the community? How are groups found to undertake new projects?

Because it is impossible to know a priori how to choose the best among alternative design choices that meet the principles, the network must provide for adequate experimentation. Prototypes of new services must be tested in small groups to determine what works well, how users will react, and the like, before final decisions are made. The network will be in a continual state of evolution from its starting configuration (which will not go away). Experimentation with new alternatives is the most efficient way to determine which growth paths are best.

5. CONCLUSIONS

A supernetwork of the magnitude of Sciencenet will remain with us for a long time. It is essential to have a clear vision of what the supernetwork will do in the long term, for otherwise it is easy to make early design decisions that will lock out important developments later.

The technology to construct a minimal Sciencenet by patching together existing networks with gateways is generally available today. This technology is

capable of little more than connections to supercomputer facilities. It does not deal with the deeper issues of uniform access over dissimilar subnetworks or of encouraging collaboration and sharing. The available technology is not good enough for the larger scientific community in the long term.

The real challenge is to come up with a design based on solid principles that will lead to a usable, robust, enduring network -- and to see the design through to a working system.

6. ACKNOWLEDGEMENTS

I appreciate the wisdom provided by Doug Comer and Frank Kuo in our recent discussions about large-network functionality. I especially appreciate comments by Barry Leiner on a draft of this paper and encouragement to turn the draft into a complete position paper. I appreciate the support of NASA through contract NAS2-11530 at the Ames Research Center.